Reproducing Nutriscore: A Machine Learning approach

Sebastian Campbell & Giridhar Ande

March 18, 2023

Reading this document

You may read this report in the format of your choosing:

HTML | PDF | GitHub

Task description

We'd like to see if we can predict Nutriscore from foods using the OpenFoodFacts API. With a suite of clustering and classification models, we'd like to work backwards to see if we can work out what determines Nutriscore, then look at the actual calculation to see how close we got. While this exercise seems pointless in itself, the same technique could be used to work backwards to calculate closed source indices, such as those used by financial industries.

We'll try a naïve approach to start with, using nutritional factors to predict nutriscore using an array of unsupervised clustering techniques to see if they form natural groups.

Afterwards, we'll use supervised classification techniques to create Nutriscore models. We'll cross-validate them and check them against a validation set.

Data description

Our initial plan was to use python `openfoodfacts` library to pull in data directly from the API. Unfortunately, the API was unreliable and had frequent down periods. We opted to use a predownloaded dump of the OpenFoodFacts data from Kaggle. The kaggle data comes in the form of a zipped tab-separated file (.tsv). When extracted, it's about 1GB in size and contains 356 027 rows and 163 columns.

Processing

In order to start using this data, we first had to filter and process it.

Nutriscore is one dimension available on OpenFoodFacts among others, including nutritional data per 100g [1]:

[1] We chose these nutrients as they're the most common ones to find. Stepping out of these moves you from 90% of foods having them, to 90% without.

- Energy (kJ)
- Protein
- Sugar
- Sodium
- Salt
- Saturated fat
- Fat
- Carbohydrates

It also contains tags for all foods, making it easy to specify food types. Calculating the Nutriscore values for everything would be tedious and take a long time to analyse so we're going to look at a few products:

- Breakfast cereals
- Iced teas[2]
- Biscuits and cakes

[2] Iced tea varies a lot in sugar content as it's often geared to the health market and so has options with either low sugar, or where some of the sugar has been replaced with artificial sweeteners.

All of these foods are highly variable in sugar and fat content so it should give us an idea of what causes Nutriscore to change.

Filtering

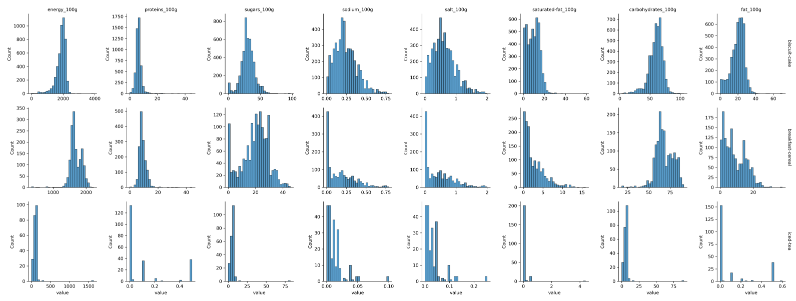
Even after reducing our dataset, we still noticed some unusual values. In particular, we noticed some salt contents that were peculiarly high.[3] We chose to remove all values with higher than 2% salt and the corresponding value for sodium.[4]

[3] Hello Panda may not be the healthiest treat, it's unlikely to contain more than its own weight in salt.

[4] Table salt is ~40% sodium, so our threshold was $(2 \times 0.4 = 0.8)$

Given that the data we had retained was relatively complete, we opted to discard any rows with missing values rather than imputing. This caused a loss of ~15% of our data.

Exploration



Unsupervised clustering

The first question we wanted to answer was about

Classification

This document demonstrates the use of a number of these page layout features to produce an attractive and usable document inspired by the Tufte handout style and the use of Tufte's styles in RMarkdown documents [@xie2018]. The Tufte handout style is a style that Edward Tufte uses in his books and handouts. Tufte's style is known for its extensive use of sidenotes, tight integration

?quarto-cite:xie2018

of graphics with text, and well-set typography. Quarto[5] supports most of the layout techniques that are used in the Tufte handout style for both HTML and LaTeX/PDF output.

5 To learn more, you can read more about Quarto or visit Quarto's Github repository.

```
---
title: "An Example Using the Tufte Style"
author: "John Smith"
format:
  html: default
  pdf: default

# places footnotes and cited sources in the margin
# other layout options (for example placing a
# figure in the margin)  will be set on per element
# in examples below
reference-location: margin
---
```

These layout features are designed with two important goals in mind:

1. To produce both PDF and HTML output with similar styles from the same Quarto document;
2. To provide simple syntax to write elements of the Tufte style such as side notes and margin figures. If you'd like a figure placed in the margin, just set the option `fig-column: margin` for your code chunk, and we will take care of the details for you[6].

6 You never need to think about `\begin{marginfigure}` or `<span class="marginfigure">`; the LaTeX and HTML code under the hood may be complicated, but you never need to learn or write such code.

If you have any feature requests or find bugs in this capabilities, please do not hesitate to file them to https://github.com/quarto-dev/quarto-cli/issues.

Figures

Margin Figures

Images and graphics play an integral role in Tufte's work. To place figures in the margin you can use the **Quarto** chunk option `column: margin`. For example:

```r
library(ggplot2)
mtcars2 <- mtcars
mtcars2$am <- factor(
  mtcars$am, labels = c('automatic', 'manual')
)
ggplot(mtcars2, aes(hp, mpg, color = am)) +
  geom_point() + geom_smooth() +
  theme(legend.position = 'bottom')
```

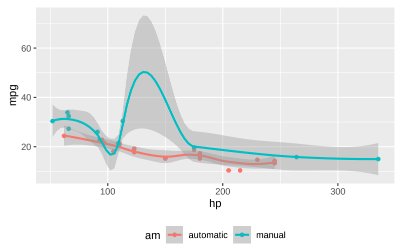


Figure 1: MPG vs horsepower, colored by transmission.